

Article

Three-Party Password Authentication and Key Exchange Protocol Based on MLWE

Songhui Guo, Yunfan Song *, Song Guo, Yeming Yang and Shuaichao Song

Third Academic, PLA Information Engineering University, Zhengzhou 450001, China; songhui.guo@outlook.com (S.G.); songguo@nudt.edu.cn (S.G.); m18888928280@163.com (Y.Y.); 14737122615@163.com (S.S.)

* Correspondence: syf1271525826@gmail.com

Abstract: With the rapid development of quantum theory, the discrete logarithm problem and significant integer factorization problem have polynomial solution algorithms under quantum computing, and their security is seriously threatened. Therefore, a three-party password-authenticated key agreement scheme based on module learning with errors problem was proposed, and its security was proved in the BPR model. Compared with other password-authenticated key agreement protocols, the proposed protocol has higher efficiency and a shorter key length, which can resist quantum attacks. Therefore, the protocol is efficient and secure and suitable for large-scale network communication.

Keywords: lattice; module learning with errors; three-party password authentication key exchange

1. Introduction

With the rapid development of internet technology and the possible arrival of quantum computers, the demand for data security has become increasingly urgent. Currently, the cryptographic algorithms based on the discrete logarithm problem and the large integer factorization problem have polynomial solving algorithms [1] under quantum computers, and their security cannot be guaranteed. Therefore, cryptographic algorithms that resist quantum computer attacks have been widely studied. Among them, the cryptographic algorithm based on lattice theory has the universality of constructing almost all cryptographic primitives and the characteristics of being able to resist quantum computer attacks, so it has become a research hotspot in the field of cryptography.

Authenticated key exchange (AKE) means that two or more participants in an open network authenticate each other and agree on a shared session key. According to different authentication methods, AKE can be divided into identity-based, certificate-based, and password-authenticated key exchange (PAKE). Password-based authenticated key exchange protocols are easy to remember and operate and can eliminate the dependence on public key infrastructure (PKI) and security hardware. Therefore, the password-authenticated key exchange (PAKE) protocol is the most widely used authenticated key exchange protocol.

The current research on the PAKE protocol mainly focuses on two-party password-authenticated key exchange (2PAKE) [2]. 2PAKE is usually based on the CS model, which requires every two participants to share a password, so the number of passwords needed to be stored will increase with the increase of the number of users communicating with it, and it is not suitable for the communication between a large number of users. Therefore, researchers proposed the three-party password-authenticated key exchange (3PAKE) [3], in which users only need to share a low-entropy password with a trusted server. The trusted server authenticates between two users and helps two users with different passwords to negotiate keys, which is suitable for large-scale network communication.

The research on the PAKE protocol mainly focuses on two-party password-based authenticated key exchange [2]. 2PAKE is usually based on the CS (client-server) model,



Citation: Guo, S.; Song, Y.; Guo, S.; Yang, Y.; Song, S. Three-Party Password Authentication and Key Exchange Protocol Based on MLWE. *Symmetry* **2023**, *15*, 1750. <https://doi.org/10.3390/sym15091750>

Academic Editor: Chin-Ling Chen

Received: 10 June 2023

Revised: 30 July 2023

Accepted: 2 August 2023

Published: 13 September 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

which requires every two participants to share a password. The complexity of passwords managed by 2PAKE increases exponentially with the number of participants, so it is unsuitable for the scenario where many participants need to mutually authenticate key exchange. For the scenario in which a large number of participants need a mutually authenticated key exchange, researchers have proposed a three-party password-authenticated key exchange protocol [3–6].

In 1995, Steiner et al. [7] first proposed the 3PAKE protocol, in which two users with different passwords authenticate each other and negotiate a key with the help of a trusted server. In the same year, Ding et al. [8] pointed out that Steiner's protocol was vulnerable to the undetectable online dictionary attack. In 2000, Lin et al. pointed out that Steiner's protocol was also vulnerable to offline dictionary attacks and proposed a new 3PAKE which can resist undetectable online dictionary attacks and offline dictionary attacks. However, its implementation depends on the server's public key and has a high communication overhead. In 2001, Lin et al. [9] improved the protocol proposed by Lin et al. [5] and proposed a 3PAKE protocol that does not rely on the server's public key, but the communication overhead is still high. In 2005, Abdalla et al. [3] proposed a security model for a 3PAKE protocol based on the BPR model and a new real-or-random (ROR) security model and constructed a general 3PAKE protocol framework based on a 2PAKE protocol. Since then, researchers have proposed several PAKE protocols [10–16] that can be provably secure in the random oracle model based on traditional mathematical problems.

Compared with the 3PAKE protocol based on traditional mathematical difficulties, the research on the 3PAKE protocol based on the lattice started late. It was not until 2013 that Ye Mao et al. [17] constructed the first 3PAKE protocol based on the lattice scheme of KATZ [18]. The scheme realizes the key exchange based on the ideal lattice and reduces the overhead of 3PAKE.

2017 Xu et al. [19] improved the 2PAKE protocol on the lattice proposed by Ding et al. [20] based on RLWE and Ding's error coordination mechanism and designed a three-party PAKE protocol. Implementation based on an error coordination mechanism is more efficient than that using the smooth projection hash function. In 2018, Wang et al. [21] improved Xu's 3PAKE protocol by considering both implicit authentication and explicit authentication. In the same year, Yu et al. [22] proposed a more efficient three-party PAKE protocol based on the split public key encryption scheme proposed by Zhang et al. [23] in 2017. In 2021, influenced by the verifier proposed by Gao [24] in 2018, Shu Qin et al. [25] designed a 3PAKE protocol based on RLWE that can resist server compromise attacks. Shu Qin et al. proved that the protocol is secure in the universally composable model.

The current PAKE protocols that can resist quantum computing attacks are mainly concentrated in two-party scenarios, and there are fewer PAKE protocols for three-party scenarios. 2PAKE can better solve the key exchange problem in the client-server environment, but it is unsuitable for communication between many users. The 3PAKE protocol can solve this problem. The current 3PAKE protocol is mainly implemented based on LWE and RLWE, and no 3PAKE protocol design is based on MLWE. MLWE is a compromise between learning with errors (LWE) and ring learning with errors (RLWE); module learning with errors (MLWE) retains the matrix format while introducing ring polynomials. Therefore, MLWE has a lower overhead than LWE and has higher security than RLWE. At the same time, MLWE can flexibly configure parameters of different security levels by adjusting the matrix dimension.

Contribution: Aiming at the current PAKE protocol that can resist quantum computing attacks, this paper constructs a three-party PAKE protocol based on the MLWE problem using Peikert error coordination technology. The new protocol proposed in this paper has the following advantages:

1. The new protocol is a three-party PAKE protocol. Compared with the two-party PAKE protocol, it can solve the problem of password storage and management in multi-user scenarios;

2. The new three-party PAKE protocol is based on the MLWE problem and the Peikert error reconciliation mechanism is implemented. Compared with the LWE problem-based scheme, its performance is better under the same security parameters. Compared with the RLWE problem-based 3PAKE, it can provide a more flexible parameter configuration;
3. The transmitted signal value of Peikert error reconciliation mechanism may bring the risk of signal leakage attack. The new three-party PAKE protocol does not need to transfer signal value in clear text, which can effectively resist signal leakage attacks.

2. Preliminaries

In this section, the definition of MLWE hard problems and the notation used in this paper are presented.

2.1. Sampling Random Variables on Lattice

Lattice cryptography adopts a particular probability distribution as noise to ensure that each sampling and the generated data are indistinguishable.

Definition 1. *Gaussian function on the lattice.*

Given the parameters $s > 0$, $c \in R^n$, the continuous Gaussian function $\rho_{s,c}(x)$ in n -dimensional space is defined as: $\rho_{s,c}(x) = \exp\left(-\frac{\pi x - c^2}{s^2}\right)$. c is called the center of the Gaussian and s is called the parameter of the Gaussian function. Then, for Gaussian function on a lattice Λ , denoted as $\rho_{s,c}(\Lambda) = \sum_{x \in \Lambda} \rho_{s,c}(x)$.

Definition 2. *The discrete Gaussian distribution on a lattice.*

It is known that the Gaussian function on the lattice Λ is $\rho_{s,c}(\Lambda) = \sum_{x \in \Lambda} \rho_{s,c}(x)$, if the random variable ξ satisfies $P_{s,c}(\xi = x) = \frac{\rho_{s,c}(x)}{\rho_{s,c}(\Lambda)}$; then, the random variable ξ is said to obey the discrete Gaussian distribution $D_{\Lambda,s,c}(x)$ with c as the center and the parameter s on the lattice.

Definition 3. *The central binomial distribution on the lattice.*

The central binomial distribution sampling on the lattice can be used to improve the sampling efficiency and ensure that the results are indistinguishable from the discrete Gaussian distribution sampling. According to the work of Bai et al. [26], vectors sampled by the central binomial distribution with parameter η are statistically indistinguishable from vectors sampled by n -dimensional discrete Gaussian distribution with parameter $\sqrt{\eta/2}$. Therefore, B_η is an approximate Gaussian distribution with zero expectation and variance $\eta/2$, which can be used as the noise distribution in learning with errors problems. When a polynomial or matrix of such polynomials is sampled from B_η in this paper, each polynomial coefficient is sampled from B_η .

Take $\{a_0, a_1, \dots, a_\eta, b_0, b_1, \dots, b_\eta\} \leftarrow \{0, 1\}^{2\eta}$ uniformly and randomly, and output $\sum_{j=1}^{\eta} (a_j - b_j)$. f is a polynomial whose coefficients satisfy the B_η distribution, and v is an n -dimensional vector composed of k polynomials f . Sample n coefficients satisfy B_η distribution from B_η distribution to form polynomial f , denoted as $f \leftarrow \beta_\eta$. When the generated k polynomials f form a vector v , it is marked as $v \leftarrow \beta_\eta^k$.

Specifically, the input of the CBD algorithm in this paper is the $n \times 2\eta$ bit output of the pseudo-random function; the output is a polynomial $f \leftarrow \beta_\eta$. The process is defined as follows Algorithm 1.

Algorithm 1: Central Binomial Distribution Sampling Algorithm

1. Input: the $n \times 2\eta$ bit sequence $B = (b_0, b_1, \dots, b_{n \times 2\eta - 1}) \in B^{n \times 2\eta}$;
2. output: polynomial $f \in R_q$.
3. For i from 0 to $n - 1$
4. $a := \sum_{j=0}^{\eta-1} b_{2i\eta+j}$,
5. $b := \sum_{j=0}^{\eta-1} b_{2i\eta+\eta+j}$,
6. $f_i := a - b$,
7. End for
8. Return $f = f_0 + f_1x + f_2x^2 + \dots + f_{n-1}x^{n-1}$

2.2. MLWE Problem

Since RLWE introduces additional algebraic structures on lattices, there may be potential security risks, such as the recently proposed attack using ring ideal lattice algebraic structures. In 2015, Langlois et al. [27] proposed the modular error-tolerant learning problem (MLWE). The design scheme introduced a tiny dimension (usually 2, 3, and 4 dimensions) in the polynomial ring structure and reduced the polynomial in the polynomial ring. The number of dimensions makes the operating efficiency comparable to RLWE while ensuring the same security. The MLWE problem can be reduced to a difficult problem on the lattice.

The following gives the definition of the MLWE problem [28] with parameters (n, q, k, η) , where n is the dimension of the vector, q is the modulus, q is the modulus in the polynomial ring $R_q^n = Z_q[x]/f(x)$, $f(x)$ is the irreducible polynomial $x^n + 1$, k is the dimension of the polynomial matrix, and β_η is a central binomial distribution on R_q^n . By randomly selecting the polynomial matrix $A \leftarrow R_q^{k \times k}$ and randomly and uniformly selecting the secret $s \leftarrow \beta_\eta^k$ and the error vector $e \leftarrow \beta_\eta^k$, $b = As + e \in R_q^k$. The following two distributions exist:

1. The distribution (A, b) , in which the polynomial matrix $A \leftarrow R_q^{k \times k}$, the secret $s \leftarrow \beta_\eta^k$, the error vector $e \leftarrow \beta_\eta^k$ is chosen uniformly at random, compute $b = As + e \in R_q^k$.
2. The distribution (A, b) , where the polynomial matrix $A \leftarrow R_q^{k \times k}$ and $b \leftarrow R_q^k$ is chosen uniformly at random.

Then, the difficult problem of search MLWE based on module lattice is given distribution 1, for (A, b) and $b = As + e \in R_q^k$ in distribution 1; finding $s \leftarrow \beta_\eta^k$ is difficult.

Then, the difficult problem of decision MLWE based on module lattice is given distribution 1 and distribution 2, and judging whether the given (A, b) is from distribution 1 or distribution 2.

2.3. Reconciliation Mechanism

The error reconciliation mechanism has a similar principle to the fuzzy extractor, which enables two parties with similar values to obtain the same value through information transmission and calculation.

By improving the original Ding-type error reconciliation mechanism [29], in 2014, Peikert [30] proposed the Peikert error reconciliation mechanism, which intercepts the high-order bits of the Z_q element so that both parties evenly extract an identical bit from each Z_q element. The specific process of the error reconciliation mechanism is described in detail below.

Supposing q is a prime number greater than 2, define $Z_q = \{-q/2, \dots, 0, \dots, q/2 - 1\}$.

Define the following three intervals: $I_0 = \{0, 1, \dots, q/4 - 1\}$, $I_1 = \{-q/4, \dots, -1\}$, $E = [-q/8, q/8) \cap Z$.

Definition 4. Error reconciliation mechanism.

The cross-rounding function: $\langle x \rangle_{q,2} = \lfloor 4/q \cdot x \rfloor \bmod 2$, if x is uniformly random, then $\langle x \rangle_{q,2}$ is uniformly random.

The modular rounding function: $\lfloor x \rfloor_{q,2} = \lfloor 2/q \cdot x \rfloor$, if x is uniformly random, then $\lfloor x \rfloor_{q,2}$ is uniformly random.

The reconciliation function: $Rec(w, \sigma) = \begin{cases} 0, w \in I_\sigma + E \bmod q; \\ 1, else \end{cases}$

When the above two functions are extended for the elements in MLWE, it is equivalent to performing the wrong reconciliation for each term of the polynomial on the R_q ring to obtain the correct shared key of n bits.

For an even number q , if $w + v = e \bmod q$ mod q is known, and $w \in Z_q$, $e \in E$, then we have $\lfloor v \rfloor_{q,2} = Rec(w, \langle v \rangle_{q,2})$.

For an odd q , then if the synchronization function rec is used directly at this time, the output is uneven. At this time, randomized function $dbl(x) = 2x - \bar{e}$ is introduced, where \bar{e} is a random term. The probability that \bar{e} is 0 is 0.5, and the probability that \bar{e} is -1 and 1 is 0.5. If $v \in Z_q$ is uniformly random, let $\bar{v} = dbl(v) \in Z_{2q}$, then given $dbl(v)$, $\lfloor \bar{v} \rfloor_{2q,2}$ on Z_{2q} is uniformly random. If $w + v = 2e \bmod 2q$ is known, and $w \in Z_{2q}$, $e \in E$, then we have $\lfloor \bar{v} \rfloor_{2q,2} = Rec(w, \langle \bar{v} \rangle_{2q,2})$.

When the above two functions are extended to the elements of a polynomial ring, R_q , it is equivalent to performing error reconciliation for each term of the polynomials in the ring to obtain the correct shared key.

2.4. PWE Assumption Based on MLWE

To facilitate the construction of the security proof of the PAKE protocol, refer to the PWE (pairing with errors) assumption proposed by Ding [20] based on RLWE hard problem and DING error reconciliation mechanism, and propose the PWE assumption based on the MLWE problem.

To determine the content of the PWE assumption based on the MLWE problem, let the adversary be an algorithm in probabilistic polynomial events, and the input (A, X, Y, σ) , where $A \leftarrow R_q^{k \times k}$, $X \leftarrow R_q^k$, $s_y, e_y, e_z \leftarrow \beta_\eta^k$, $Y = A^T s_y + e_y \in R_q^k$, $\sigma = dbl(X^T s_y + e_z)_{2q,2} \in \{0, 1\}^n$, $K = dbl(X^T s_y + e_z)_{2q,2} \in \{0, 1\}^n$. Then, the goal of the adversary C is to obtain the value of the K from the input of (A, X, Y, σ) . In this paper, the adversary's advantage in breaking the PWE assumption based on MLWE problem is formally defined as follows.

$$Adv_{R_q}^{MPWE}(C) = \Pr \left[\begin{array}{l} A \leftarrow R_q^{k \times k}; s_y \leftarrow \beta_\eta^k; e_y \leftarrow \beta_\eta^k; e_z \leftarrow \beta_\eta^k; Y \leftarrow A^T s_y + e_y; \\ \sigma \leftarrow dbl(X^T s_y + e_z)_{2q,2} : K = C(A, X, Y, \sigma) \end{array} \right]$$

Let $Adv_{R_q}^{MPWE}(t, N) = \max_C \{ Adv_{R_q}^{MPWE}(C) \}$, where, all adversaries with the maximum time complexity t will take advantage of the maximum attack, and these adversaries will output a list containing at most N elements belonging to $\{0, 1\}^n$. The MPWE assumption shows that t is negligible for t and N under the bounds of security parameters. The decision version of the MPWE problem can be defined as follows.

Definition 5. Decision MPWE problem.

Given $(A, X, Y, \sigma, K) \in R_q^{k \times k} \times R_q^k \times R_q^k \times \{0, 1\}^n \times \{0, 1\}^n$, where $\sigma = dbl(X^T s_y + e_z)_{2q,2}$, $K = dbl(X^T s_y + e_z)_{2q,2} \in \{0, 1\}^n$. Set $Z = X^T s_y + e_z$, then the DMPWE problem is to decide whether $Y \in R_q^k$ and randomly uniformly generated $Y' \leftarrow R_q^k$ can be distinguished. If DMPWE is hard, then MPWE is hard.

Before reducing the DMPWE problem to the MLWE problem, it is first necessary to define an MLWE-DH problem, which can be reduced to the decision MLWE problem.

Definition 6. *MLWE-DH problem.*

Given $(A, X, Y, Z) \in R_q^{k \times k} \times R_q^k \times R_q^k \times R_q$, where $A \leftarrow R_q^{k \times k}$, $X \leftarrow R_q^k$, then the MLWE-DH problem is to decide whether $(Z = X^T s_y + e_z, Y \leftarrow A^T s_y + e_y) \in R_q \times R_q^k$ and random uniformly generated $(Z', Y') \leftarrow R_q \times R_q^k$ can be distinguished.

Theorem 1. *Assuming that the decision MLWE problem is hard, the MLWE-DH problem is also hard.*

Proof of Theorem 1. Suppose there exists an algorithm D that can solve the MLWE-DH problem with a non-negligible advantage on the input (A, X, Y, Z) . Then an algorithm D' can be constructed to solve the decision MLWE problem based on the algorithm. Specifically, two decision MLWE instances (A_1, b_1) and (A_2, b_2) with the same private key $s_y \in R_q^k$ are first given. The execution process of the algorithm D' is as follows:

1. Set $(A, X, Y, Z) = (A_1, A_2, b_1, b_2)$;
2. Input (A, X, Y, Z) into the algorithm D ;

If the algorithm D outputs 1, then that means $(Z = X^T s_y + e_z, Y \leftarrow A^T s_y + e_y) \in R_q \times R_q^k$, $(b_2 = A_2^T s_y + e_z, b_1 \leftarrow A_1^T s_y + e_y) \in R_q \times R_q^k$, the algorithm D' also outputs 1. If the algorithm D outputs 0, then it means that $(Z, Y) \in R_q \times R_q^k$ is randomly uniformly generated. Therefore, $(b_2, b_1) \in R_q \times R_q^k$ is randomly uniformly generated, in which case the algorithm D' also outputs 0. The decision MLWE problem is solved. \square

Suppose an algorithm can solve the MLWE-DH problem with non-negligible advantage. In that case, one can construct an algorithm to solve the decision MLWE problem with non-negligible advantage. However, this is contrary to the hardness of the decision MLWE problem itself, so if the decision MLWE problem is hard, the MLWE-DH problem is also hard.

Theorem 2. *Assuming that the MLWE-DH problem is hard, the DMPWE problem is also hard.*

Proof of Theorem 2. Suppose an algorithm D exists that can solve the DMPWE problem with a non-negligible advantage on the input (A, X, Y, σ, K) . An algorithm D' can be constructed based on the algorithm D to solve the MLWE-DH problem. Specifically, the algorithm D' execution process is as follows:

1. For an instance of the MLWE-DH problem (A, X, Y, Z) , set $\sigma = \text{dbl}(Z)_{2q,2} \in \{0, 1\}^n$, $K = \text{dbl}(Z)_{2q,2} \in \{0, 1\}^n$;
2. Input (A, X, Y, σ, K) to the algorithm D ;

If the algorithm D outputs 1, then that means $(Y \leftarrow A^T s_y + e_y) \in R_q^k$. Therefore, $(b_1 \leftarrow A^T s_y + e_y) \in R_q^k$, the algorithm D' also outputs 1. If the algorithm D outputs 0, then it means that $Y \in R_q^k$ is randomly uniformly generated. Therefore, $b_1 \in R_q^k$ is randomly uniformly generated, in which case the algorithm D' also outputs 0. The MLWE-DH problem is solved. \square

Suppose an algorithm D can solve the DMPWE problem with a non-negligible advantage. In that case, an algorithm D' can be constructed to solve the MLWE-DH problem with non-negligible advantage.

It can be seen from Theorem 1 that the MLWE-DH problem is difficult, which is contrary to the solvability of the MLWE-DH problem. Therefore, if the MLWE-DH problem is hard, then DMPWE is also hard.

2.5. Three-Party PAKE Security Model

This section mainly adopts [20,31,32] models in the literature to give a more realistic three-party PAKE security model, which can more accurately evaluate the real risk faced by the three-party PAKE protocol.

Security game: an algorithm game between the challenger C and the adversary. The challenger C runs an instance of the simulator simulating the protocol P , running the protocol P on behalf of the honest user.

System model: Similar to the existing three-party PAKE protocol, the system model consists of a server set and a user set, while also assuming the existence of a dictionary library of size $|D|$. At the protocol's beginning, two users and are extracted from the user set, and their passwords PW_i and passwords PW_j are assigned from the dictionary library, respectively. Then, extract a server from the server set and assign it a password PW_i and a password PW_j . The users U_i and U_j authenticate and exchange information to establish a shared session key through the server.

Adversary capability: An adversary A operating in probabilistic polynomial time is assumed to have complete control over the communication channel. The adversary A can create, forward or modify a message. The adversary can also create multiple instances to participate in the concurrent execution of the protocol. The participant instance of the 3PAKE protocol is denoted by identity Π_p^i ; the two instances of the user are denoted by Π_A^i and Π_B^i , and the instance of the server is denoted by Π_S^i . The security of the protocol is defined by a series of games between the challenger C and the adversary A , in which the adversary A can query any given participant instance as follows:

Execute ($\Pi_A^i, \Pi_B^i, \Pi_S^i$) query: This query characterizes the passive attack capability of the adversary A . The adversary can obtain the information transmitted over the channel during the honest interaction of the protocol by querying the instances.

Send (Π_p^i, m) query: This query describes the dynamic attack capability of the adversary A . The adversary can interact with the Π_p^i instance by intercepting, forwarding, and modifying the generated information m . The oracle output is the reply message executed according to the protocol specification after Π_p^i receiving the generated information from the adversary A .

Reveal (Π_U^i) query: This query captures the ability to leak session keys when a participant in a user instance Π_U^i misuses a session key. The adversary A obtains the session key SK_U^i in the user instance through this query.

Corrupt (Π_p^i) query: This query characterizes the forward security of the protocol, allowing the adversary A to damage the protocol participants at will. If the participant is a server, return the passwords of both users to the adversary A ; if the participant is a user, return the passwords of the corresponding users to the adversary A .

Test (Π_p^i) query: This query characterizes the adversary's ability to distinguish a real session key from random values. Randomly flip a coin $b \in \{0, 1\}$; if $b = 0$, return the real negotiated key, if $b = 1$, return any random number in the session key space. Finally, the adversary A outputs b' ; if $b' = b$, the adversary A wins. If the result of this query is guaranteed to be valid, it should be assured that the queried session instance is fresh.

Based on the above description, this section gives the following definitions:

Definition 7. Partnership.

This article uses session identifiers to define partnerships, where Sid is an identifier used to uniquely name one of the sessions corresponding to that instance. Pid is used to determine the identity of the user instance that is talking to the instance. SK is the value of the shared secret that both user instances Π_A^i and Π_B^i have completed the calculation of the last step stipulated in the protocol. Both Π_A^i and Π_B^i instances maintain list $(Sid_A^i, Pid_A^i, SK_A^i)$ and $(Sid_B^i, Pid_B^i, SK_B^i)$, respectively. Partnership is said to be satisfied when the following conditions are satisfied: 1. $Sid_A^i = Sid_B^i$; 2. $Pid_A^i = B, Pid_B^i = A$; 3. $SK_A^i = SK_B^i$.

Definition 8. *Freshness.*

Π_A^i is a fresh instance when Π_A^i is an honest session, and satisfies the following conditions:

1. The adversary has not performed the Reveal (Π_U^i) query on the user instance Π_A^i or its partner Π_B^i ;
2. The adversary has not executed the Corrupt (Π_P^i) query on the user instance Π_A^i , the partner instance Π_B^i , or the server instance Π_S^i .

Definition 9. *Semantic safety.*

The PAKE protocol is secure if the following conditions are satisfied:

Two honest user instances satisfy the partnership and compute the same session key (otherwise the protocol fails).

Under the BPR model, the goal of adversary A is to identify the real session key from a given random key and the real session key. The adversary performs a special test (Π_P^i) query. Finally, adversary A outputs b' , and if $b' = b$, then adversary A wins. The advantage of adversary A in attacking instance Π_U^i is defined as: $Adv_{\Pi}^{A,ke}(A) = |2\Pr(b' = b) - 1|$. The 3PAKE protocol is semantically safe if $Adv_{\Pi}^{A,ke}(A)$ is negligible for all probabilistic polynomial-time adversaries A .

3. Our Protocol

Aiming at the current PAKE protocol that can resist quantum computing attacks, this chapter constructs a three-party PAKE protocol based on the MLWE problem and the Peikert error reconciliation technique. The new protocol has the following advantages: (1) The new protocol is a three-party PAKE protocol, compared with the two-party PAKE protocol, so it can solve the problem of password storage and management in multi-user scenarios. (2) The new three-party PAKE protocol is implemented based on the MLWE problem and Peikert error reconciliation mechanism, which has better performance than the scheme based on the LWE problem under the same security parameters and can provide more flexible parameter configuration than the 3PAKE based on RLWE problem. (3) The signal value transmitted by the Peikert error reconciliation mechanism may bring the risk of signal leakage attack. The new tripartite PAKE protocol does not need to transmit the signal value in plaintext, which can effectively resist the signal leakage attack.

3.1. System Initialization Process

In the PAKE scheme in this paper, R and R_q represent $Z[x]/(x^n + 1)$ and $Z_q[x]/(x^n + 1)$, respectively; the lowercase italic letters represent column vectors, in which the elements belong to the ring R or R_q ; the capital italic letters indicate matrix. Let (n, q, k) be the public parameters of the PAKE scheme, where n is a power of 2, and q is an odd prime and satisfies $q \equiv 1 \pmod{n}$. $A \leftarrow R_q^{k \times k}$ represents a sample matrix uniformly from $R_q^{k \times k}$. $(s, e) \leftarrow \beta_{\eta}^k \times \beta_{\eta}^k$ represents a sample private key and noise uniformly from β_{η}^k .

3.2. Registration Process

When users i join the system, they need to register with the server S through a secure channel. The details are as follows:

1. $i \rightarrow S : (ID_i, PW_1)$. The user i selects an identity ID_i , a private password PW_0 , and calculates shared password $PW_1 = H_0(ID_i || PW_0)$. After that, the user i sends registration request (ID_i, PW_1) to the server S .
2. $S \rightarrow i : (TID_i, H_1(\cdot))$. The server S receives the registration request (ID_i, PW_1) from the user i , generates random number n_0 , calculates the temporary identity $TID_i = H_0(ID_i || n_0 || PW_1)$ of the user i , and retains the shared password PW_1 . After that, the server S sends TID_i to the user i .

3.3. Authentication and Key Exchange Phase

When users A and users B need to share the session key for communication, it is necessary to establish two-way authentication between users A with users B based on the trusted server to ensure the legitimacy of the identities of both users A and users B , and an exchange shared session key.

The execution process of the protocol is described in Figure 1:

$$1. \quad A \rightarrow S : M_0 = (TID_A, TID_B, n_1, b_{A0})$$

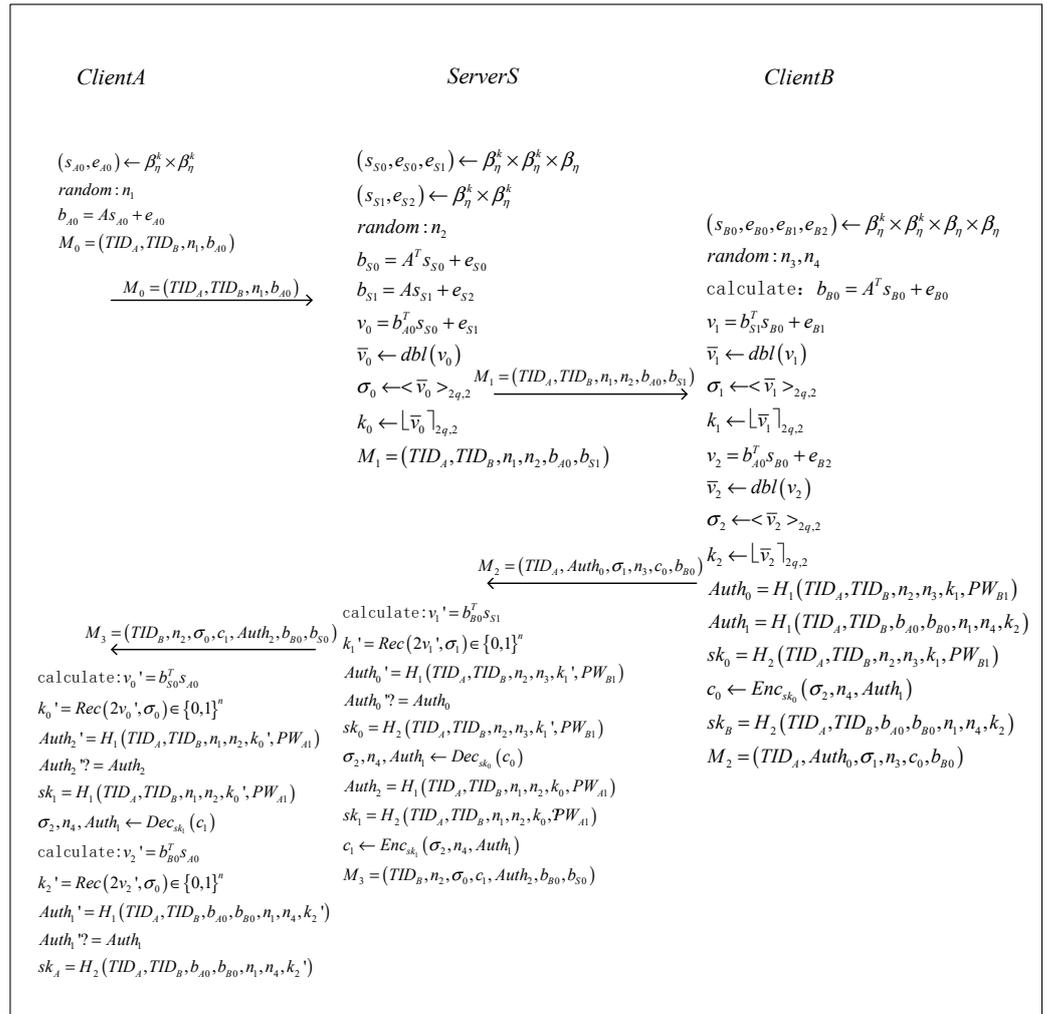


Figure 1. Authentication and key exchange phase.

The user A samples $(s_{A0}, e_{A0}) \leftarrow \beta_\eta^k \times \beta_\eta^k$ uniformly, generates random number n_1 , and calculates $b_{A0} = As_{A0} + e_{A0}$. The user A sends $M_0 = (TID_A, TID_B, n_1, b_{A0})$ to the server S .

$$2. \quad S \rightarrow B : M_1 = (TID_A, TID_B, n_1, n_2, b_{A0}, b_{S1})$$

Upon the server receiving M_0 , the server confirms the identity of the client applying for authentication and key exchange through (TID_A, TID_B) . Then, the server samples $(s_{S0}, e_{S0}, e_{S1}) \leftarrow \beta_\eta^k \times \beta_\eta^k \times \beta_\eta$ and $(s_{S1}, e_{S2}) \leftarrow \beta_\eta^k \times \beta_\eta^k$ uniformly generate random number n_2 and calculate $b_{S0} = A^T s_{S0} + e_{S0}$, $b_{S1} = As_{S1} + e_{S2}$ and $v_0 = b_{A0}^T s_{S0} + e_{S1}$. The server uses the randomized function, cross-rounding function, and modular rounding function to calculate b . The randomized function, cross-rounding function, and modular rounding function are used to calculate $\bar{v}_0 \leftarrow dbl(v_0)$, $\sigma_0 \leftarrow \langle \bar{v}_0 \rangle_{2q,2}$ and $k_0 \leftarrow \lfloor \bar{v}_0 \rfloor_{2q,2}$, and send $M_1 = (TID_A, TID_B, n_1, n_2, b_{A0}, b_{S1})$ to the user B .

3. $B \rightarrow S : M_2 = (TID_A, Auth_0, \sigma_1, n_3, c_0, b_{B0})$

After the user B receives the message M_1 , user B sample $(s_{B0}, e_{B0}, e_{B1}, e_{B2}) \leftarrow \beta_{\eta}^k \times \beta_{\eta}^k \times \beta_{\eta} \times \beta_{\eta}$ uniformly generates random numbers n_3, n_4 , and calculates $b_{B0} = A^T s_{B0} + e_{B0}$, $v_1 = b_{S1}^T s_{B0} + e_{B1}$ and $v_2 = b_{A0}^T s_{B0} + e_{B2}$. The user B uses the randomized function to calculate $\bar{v}_1 \leftarrow dbl(v_1)$ and $\bar{v}_2 \leftarrow dbl(v_2)$; uses the cross-rounding function to calculate $\sigma_1 \leftarrow \langle \bar{v}_1 \rangle_{2q,2}$ and $\sigma_2 \leftarrow \langle \bar{v}_2 \rangle_{2q,2}$; uses the modular rounding function to calculate $k_1 \leftarrow \lfloor \bar{v}_1 \rfloor_{2q,2}$ and $k_2 \leftarrow \lfloor \bar{v}_2 \rfloor_{2q,2}$.

The user B calculates the authentication information $Auth_0 = H_1(TID_A, TID_B, n_2, n_3, k_1, PW_{B1})$ between the server and user B , and calculates the authentication information $Auth_1 = H_1(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_3, n_4, k_2)$ between user A and user B . Then, user B calculates the shared key $sk_0 = H_2(TID_A, TID_B, n_2, n_3, k_1, PW_{B1})$ between user B and uses the shared key to encrypt $(\sigma_2, n_4, Auth_1)$. At this time, user B sets the shared key $sk_B = H_2(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_3, n_4, k_2)$ between user B and user A . User B sends $M_2 = (TID_A, Auth_0, \sigma_1, n_3, c_0, b_{B0})$ to the server.

4. $S \rightarrow A : M_3 = (TID_B, n_2, \sigma_0, c_1, Auth_2, b_{B0}, b_{S0})$

The server calculates $v_1' = b_{B0}^T s_{S1}$, then the user uses the Peikert error reconciliation function to calculate $k_1' = Rec(2v_1', \sigma_1) \in \{0, 1\}^n$. The server calculates the verification information $Auth_0' = H_1(TID_A, TID_B, n_2, n_3, k_1', PW_{B1})$ with user B and the verification information $Auth_2 = H_1(TID_A, TID_B, n_1, n_3, k_0, PW_{A1})$ with user A . If $Auth_0' = Auth_0$, the server continues to calculate the shared key $sk_0 = H_2(TID_A, TID_B, n_2, n_3, k_1', PW_{B1})$ with user B and the shared key $sk_1 = H_2(TID_A, TID_B, n_1, n_2, k_0, PW_{A1})$ with user A , and the server uses the shared key sk_0 to decrypt c_0 to obtain $\sigma_2, n_4, Auth_1$. After the user finishes decrypting, the server uses the shared key sk_1 with user A to encrypt $\sigma_2, n_4, Auth_1$ to obtain c_1 . The server sends $M_3 = (TID_B, n_2, \sigma_0, c_1, Auth_2, b_{B0}, b_{S0})$ to user A .

5. $sk_A = sk_B$

After user A receives the message M_3 , user A calculates $v_0' = b_{S0}^T s_{A0}$ and uses the Peikert error reconciliation function to calculate $k_0' = Rec(2v_0', \sigma_0)$. Then, user A calculates $Auth_2' = H_1(TID_A, TID_B, n_1, n_2, k_0', PW_{A1})$ and compares it with the received authentication information $Auth_2$ of the server; if $Auth_2' = Auth_2$, the server's identity is credible. User A calculates the shared key $sk_1 = H_2(TID_B, n_1, n_2, k_0', PW_{A1})$ with the server, and decrypts c_1 to obtain $\sigma_2, n_4, Auth_1$. Based on the decrypted information, user A continues to calculate $v_2' = b_{B0}^T s_{A0}$ and uses the Peikert error reconciliation function to calculate $k_2' = Rec(2v_2', \sigma_2)$. Then, the user calculates $Auth_1' = H_1(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_4, k_2')$ and compares it with $Auth_1$. If $Auth_1' = Auth_1$, user A 's identity is credible, and user A obtains the same session key $sk_A = H_2(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_4, k_2')$ as user B .

4. Proof of Correctness of the Protocol

If the protocol participants all run the protocol honestly, they will obtain $sk_A = sk_B$ with significant probability. In the protocol, the following is the correctness proof that user A and user B obtain the same session key when the honest user executes the scheme.

It is known that $sk_A = H_2(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_4, k_2')$ and $sk_B = H_2(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_4, k_2)$.

So, if $k_2' = k_2$, user A and user B will obtain the same session key. This paper has $k_2' = Rec(2v_2', \sigma_2)$ and $k_2 \leftarrow \lfloor \bar{v}_2 \rfloor_{2q,2}$. From the Peikert error reconciliation function, if all the coefficients of the polynomial obtained by $(\bar{v}_2 - 2v_2')$ are not in $[-q/4, q/4]$, then $k_2' = k_2$ can be obtained in this paper.

$$\bar{v}_2 = dbl(v_2) = 2v_2 + \bar{e} \quad (1)$$

$$v_2 = b_{A0}^T s_{B0} + e_{B2} = (As_{A0} + e_{A0})^T s_{B0} + e_{B2} = s_{A0}^T A^T s_{B0} + e_{A0}^T s_{B0} + e_{B2} \quad (2)$$

$$v_2' = b_{B0}^T s_{A0} = \left(A^T s_{B0} + e_{B0} \right)^T s_{A0} = s_{B0}^T A s_{A0} + e_{B0}^T s_{A0} = \left(s_{A0}^T A^T s_{B0} \right)^T + e_{B0}^T s_{A0} \quad (3)$$

Since $s_{A0}^T A^T s_{B0}$ is a polynomial, so $\bar{v}_2 - 2v_2' = 2e_{A0}^T s_{B0} + 2e_{B2} + \bar{e} - 2e_{B0}^T s_{A0}$. According to the central binomial distribution sampling algorithm, all the coefficients of the obtained polynomial $(2e_{A0}^T s_{B0} + 2e_{B2} + \bar{e} - 2e_{B0}^T s_{A0})$ are not in $[-q/4, q/4]$. So, from the Peikert error reconciliation function, user A and user B will obtain $sk_A = sk_B$ with significant probability.

5. Security Analysis

This section proves the AKE security of the three-party PAKE protocol proposed in this paper based on the BPR model, and analyzes the security properties satisfied by the protocol against known attacks.

5.1. Security proof

This section is mainly based on the BPR model to prove the security of the three-party PAKE protocol scheme proposed in this paper. Each participant, including the adversary, is simulated in this protocol section as a set of probabilistic polynomial oracles. Suppose there is a polynomial time adversary A , user instances Π_A^i and Π_B^i of the i th session, and server instance Π_S^i . The adversary's ability can be abstracted as several queries on the Execute, Send, Reveal, Corrupt, and Test oracles.

Theorem 3. Let Π be the protocol proposed in this paper, and D be the dictionary library of size $|D|$. q_{se} , q_{ex} , q_{re} , q_{co} respectively represent Send, Execute, Reveal, Corrupt queries, and q_{ro} represent querying random oracles. Assume that the emulator controls all oracles that the adversary A has access to. The simulator runs the protocol Π , including choosing a password for each user. Then, for a polynomial time adversary A , the advantage of its attack protocol is defined as: $Adv_{\Pi}^{Ake}(A) \leq (q_{ro} + q_{se} + q_{ex})^2 / q^{nk} + q_{se} / 2^d + q_{ro} / q^{nk} + 2Adv_{R_q}^{MPWE}(t', q_{ro}) + 2Adv_{R_q}^{DMLWE}(t', q_{ro}) + q_{se} / |D|$.

Proof of Theorem 3. The game $G_i (i = 0, \dots, 7)$ is defined in the security model of this section. Among these, $G_0 = \Pi$ is equivalent to the honest implementation of the protocol. In G_7 , it is equivalent to simulating the protocol in the ideal situation under the random oracle model; at this time, the advantage of the adversary attacking the protocol is negligible. For any game G_i , define the event that the adversary guesses correctly to randomly select the bit b in the Test query as $Succ_i$. The advantage of the adversary attacking the protocol in G_i is greater than that of the adversary attacking the protocol in G_{i-1} ; that is, the security of the protocol is gradually reduced, so we can obtain:

$$Adv_{G_0}^{Ake}(A) \leq Adv_{G_1}^{Ake}(A) + \varepsilon_1 \leq Adv_{G_2}^{Ake}(A) + \varepsilon_2 \leq Adv_{G_3}^{Ake}(A) + \varepsilon_3 \leq Adv_{G_4}^{Ake}(A) + \varepsilon_4 \leq Adv_{G_5}^{Ake}(A) + \varepsilon_5 \leq Adv_{G_6}^{Ake}(A) + \varepsilon_6 \leq Adv_{G_7}^{Ake}(A) + \varepsilon_7 \quad (4)$$

The $\varepsilon_1 \dots \varepsilon_7$ in the formula is negligible. By combining these negligible values with the probability of success of an online password-guessing attack, this paper can calculate the adversary's advantage of the success in attacking the protocol.

For ease of understanding, this article distinguishes between user queries Π_A^i , user queries Π_B^i , and server queries Π_S^i . Adversary A makes one of the following queries:

A0 query: whether to instruct some unused Π_A^i instance to send the first message to the server instance Π_S^i , which corresponds to the user Π_A^i start of the authenticated key exchange phase;

S1 query: whether some messages were sent to a previously unused server instance Π_S^i , and the server instance Π_S^i is expected to send some messages to user instance

Π_B^i , which corresponds to the first response of the server during the authentication key exchange phase;

B1 query: whether a message has been sent to an unused user instance Π_B^i , and the user instance is expected to send a message to the server instance Π_S^i , which corresponds to the first response of the user Π_B^i in the authentication key exchange phase;

S2 query: whether a message was sent to a used server instance Π_S^i and is expected to send a message to a user instance Π_A^i , which corresponds to the second response of the server during the authentication key exchange phase;

A1 query: Whether a message was sent to a user instance Π_A^i indicating that this is the last message for this key exchange, corresponding to the last message received during the authenticated key exchange phase.

For ease of understanding, the adversary can define session key guessing and password guessing events for user instances and server instances at any stage in the query process:

Testsk (A, i, B, S, l) : For $b_{A0}, b_{B0}, b_{S0}, b_{S1}$, adversaries A perform a query $H_l(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_4, k_2')$; the query A0, whose output is $(TID_A, TID_B, n_1, b_{A0})$, the A1 query whose input is $(TID_B, n_2, \sigma_0, c_1, Auth_2, b_{B0}, b_{S0})$, and the nearest query is the query $H_l(\cdot)$ or query A1, where $k_2' = Rec(2v_2', \sigma_2)$ and $v_2' = b_{B0}^T s_{A0}$, the event's associated value is the output of $H_l(\cdot), l \in \{1, 2\}$ (representing $Auth_1', sk_A$, respectively).

Testsk! (A, i, B, S, l) : For b_{B0}, c_1 , making a A1 query with input $(TID_B, n_2, \sigma_0, c_1, Auth_2, b_{B0}, b_{S0})$ and results in the event Testsk $(A, i, B, S, 1)$ with associated value is $Auth_1'$.

Testsk (B, j, A, S, l) : For $b_{A0}, b_{B0}, b_{S0}, b_{S1}$, the adversary has made B1 queries with the input $(TID_A, TID_B, n_1, n_2, b_{A0}, b_{S1})$ and output $(TID_A, Auth_0, \sigma_1, n_3, c_0, b_{B0})$. Then, the adversary has made the $H_l(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_4, k_2)$ query, where $k_2 \leftarrow [\bar{v}_2]_{2q,2}$, $v_2 = b_{A0}^T s_{B0} + e_{B2}$. The event's associated value is the output of $H_l(\cdot), l \in \{1, 2\}$ (representing $Auth_1', sk_A$ respectively).

Testsk* (B, j, A, S) : For $l \in \{1, 2\}$, Testsk (B, j, A, S, l) occurs.

Testsk (A, i, B, j, S) : For $l \in \{1, 2\}$, the Testsk (A, i, B, S, l) and Testsk (B, j, A, S, l) events occur simultaneously, where Π_A^i is paired with Π_B^j and Π_B^j is paired with Π_A^i after the S2 query.

Testexcesk (A, i, B, j, S) : For $b_{A0}, b_{B0}, b_{S0}, b_{S1}$, and adversary A has executed $(\Pi_A^i, \Pi_B^j, \Pi_S^i)$ query with output $(b_{A0}, b_{B0}, b_{S0}, b_{S1}, c_1)$, then adversary A has made $H_l(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_4, k_2)$, where $k_2 \leftarrow [\bar{v}_2]_{2q,2} = Rec(2v_2', \sigma_2)$. The event's associated value is the output of $H_l(\cdot), l \in \{1, 2\}$ (representing $Auth_1', sk_A$, respectively).

Correctsk: Testsk! (A, i, B, S, l) events occurs on A, i, B and S or Testsk* (B, j, A, S) occurs on B, j, A , and S , before any Corrupt queries.

Correctskexec: Testexcesk (A, i, B, j, S) occurs on A, i, B, j, S .

Pairedskguess: For A, i, B, j, S , the Testsk (A, i, B, j, S) event occurs.

Correctauth 0: For b_{B0}, b_{S1} , the adversary makes the $H_1(TID_A, TID_B, n_2, n_3, k_1', PW_{B1})$ query, the S1 query with input $(TID_A, TID_B, n_1, b_{A0})$ and output $(TID_A, TID_B, n_1, n_2, b_{A0}, b_{S1})$, and S2 query with input as $(TID_A, Auth_0, \sigma_1, n_3, c_0, b_{B0})$, where the nearest query is the $H_1(\cdot)$ query or S2 query. If $k_1' = Rec(2v_1', \sigma_1)$, $v_1' = b_{B0}^T s_{S1}$, the associated value for this event is $Auth_0$.

Correctauth 2: For b_{A0}, b_{S0} , the adversary makes the $H_1(TID_A, TID_B, n_1, n_2, k_0', PW_{A1})$ query, the A0 query with input $(TID_A, TID_B, n_1, b_{A0})$, the A1 query with input $(TID_B, n_2, \sigma_0, c_1, Auth_2, b_{B0}, b_{S0})$, and the nearest query is the $H_1(\cdot)$ query or the A1 query. $k_0' = Rec(2v_0', \sigma_0)$, $v_0' = b_{S0}^T s_{A0}$. The associated value for this event is $Auth_2$.

Correctpw: The adversary made a correct guess about the user's password.

The next step is to make security reduction on the 3PAKE protocol proposed in this paper:

G_0 : This game simulates a real attack under the random oracle model for the PAKE protocol proposed in this paper. The advantage of an adversary in breaking the protocol can be defined as: $Adv_{G_0}^{Ake}(A) = 2Pr[Succ] - 1$.

G_1 : In this game, the simulator simulates a random oracle $H_i(\cdot)$ ($i \in \{0, 1\}$) by maintaining hash lists $\wedge H_0$ and $\wedge H_1$. In a hash query, if there is a record (m, r) in the hash list, r is returned; otherwise, an element $r \in Z_q$ is randomly selected, (m, r) is added to the list, and r is returned. \square

Lemma 1. For probabilistic polynomial adversaries A , G_1 and G_0 are indistinguishable.

Proof. Obviously, unless adversary A can break the one-way hash function, adversary A cannot distinguish the output of the hash function and random string. Thus, G_1 and G_0 are indistinguishable: $|Adv_{G_0}^{Ake}(A) - Adv_{G_1}^{Ake}(A)| \leq negl$.

G_2 : G_2 and G_1 are indistinguishable unless an honest player randomly chooses $b_{A0}, b_{B0}, b_{S0}, b_{S1}$ which appeared in a previous query; then, the protocol aborts and the adversary fails. \square

Lemma 2. For probabilistic polynomial adversaries A , the advantage of distinguishing between games G_2 and G_1 games is: $|Adv_{G_2}^{Ake}(A) - Adv_{G_1}^{Ake}(A)| \leq (q_{ro} + q_{se} + q_{ex})^2 / q^{nk}$.

Proof. That is, b_{A0} cannot be equal to the b_{A0} that appeared in the previous Execute $(\Pi_A^i, \Pi_B^i, \Pi_S^i)$, Send (Π_P^i, m) query, A0 query, S1 query or B1 query and random oracle query; b_{S0} cannot be equal to b_{S0} that appeared in the previous Execute $(\Pi_A^i, \Pi_B^i, \Pi_S^i)$, Send (Π_P^i, m) query, S2 query, A1 query, or random oracle query. b_{S1} cannot be equal to the b_{S1} that appears in the previous Execute $(\Pi_A^i, \Pi_B^i, \Pi_S^i)$, Send (Π_P^i, m) query, S1 query, B1 query, or random oracle query; b_{B0} cannot be the same as b_{B0} that appeared in the previous Execute $(\Pi_A^i, \Pi_B^i, \Pi_S^i)$, Send (Π_P^i, m) query, B1 query, S2 query, A1 query, or random oracle query.

From the birthday attack, the probability of distinguishing G_2 from G_1 for a probabilistic polynomial adversary does not exceed: $|Adv_{G_2}^{Ake}(A) - Adv_{G_1}^{Ake}(A)| \leq (q_{ro} + q_{se} + q_{ex})^2 / q^{nk}$.

G_3 : G_3 is same as G_2 , except that the adversary does not use the random oracle model in the output of Execute $(\Pi_A^i, \Pi_B^i, \Pi_S^i)$ and Send (Π_P^i, m) queries. Subsequent oracle queries by the adversary will be as consistent as possible with Execute $(\Pi_A^i, \Pi_B^i, \Pi_S^i)$ and Send (Π_P^i, m) queries. The specific queries are answered as follows:

For the Execute $(\Pi_A^i, \Pi_B^i, \Pi_S^i)$ query, $b_{A0} = As_{A0} + e_{A0}$, $b_{S0} = A^T s_{S0} + e_{S0}$, $b_{S1} = As_{S1} + e_{S2}$, $b_{B0} = A^T s_{B0} + e_{B0}$, where $s_{A0}, e_{A0}, s_{S0}, e_{S0}, s_{S1}, e_{S2}, s_{B0}, e_{B0}$ is taken randomly from the distribution β_η^k , $Auth_0, \sigma_1$, $Auth_2$ random uniform is taken from the distribution $\{0, 1\}^d$, and c_0, c_1 is generated uniformly at random from distribution $\{0, 1\}$.

For S1 queries on server instances Π_S^i , $b_{A0} = As_{A0} + e_{A0}$, where $s_{A0}, e_{A0}, s_{S1}, e_{S2}$ are randomly taken from the distribution β_η^k .

For B1 queries on the user instance Π_B^i , $b_{B0} = A^T s_{B0} + e_{B0}$ where s_{B0}, e_{B0} is taken randomly from the distribution β_η^k , and $Auth_0, \sigma_1$ are taken randomly uniformly from the distribution $\{0, 1\}^d$, and c_0 is the generated uniformly at random from distribution $\{0, 1\}$.

For S2 queries on server instances Π_S^i , if the query results in a Correctauth0 event, set $b_{S0} = A^T s_{S0} + e_{S0}$, where s_{S0}, e_{S0} are randomly taken from the distribution β_η^k . Otherwise, the server instance aborts.

For A1 queries on the user instance Π_A^i , if the query results in a Correctauth2 event, perform the following steps:

If the user instance Π_A^i is not yet paired with a user instance Π_B^i , and the query results in a Testsk! (A, i, B, S, l) , then set sk_A to the relevant value of Testsk $(A, i, B, S, 2)$.

Set $sk_A = sk_B$, if the user instance Π_A^i has already paired with a user instance Π_B^i .

Otherwise, the user instance Π_A^i terminates.

If the query does not result in a Correctauth2 event, the user instance Π_A^i terminates.

For $H_l(\cdot)$ query, $l \in \{1, 2\}$, if the query results in the occurrence of the events Testsk (A, i, B, S, l) , Testsk (B, j, A, S, l) , Testexcsk (A, i, B, j, S) , Correctauth0, or Correctauth2, then output the associated value of the event, otherwise output a random value. \square

Lemma 3. For any polynomial adversary, $\left| Adv_{G_3}^{Ake}(A) - Adv_{G_2}^{Ake}(A) \right| \leq q_{se}/2^d + q_{ro}/q^{nk}$.

Proof. The design of G_3 is a standard technique used in the security analysis of random oracle. G_3 and G_2 are indistinguishable unless the adversary makes the following two queries:

The Correctauth0 or Correctauth2 event caused by the $H_l(\cdot)$ query with the correct password as input, but the total probability of this happening is at most q_{ro}/q^{nk} since the adversary cannot actually obtain the correct password.

The Send (Π_p^i, m) query terminates either the user instance or the server instance. According to the above analysis, the query that causes the termination of the user instance Π_A^i is the A1 query, and the query that causes the termination of the server instance Π_S^i is the S2 query. If the query does not cause a Correctauth0 event, the server instance terminates. The probability of termination is not more than $q_{se}/2^d$.

Thus, for any polynomial adversary, the advantage of distinguishing G_3 and G_2 is $\left| Adv_{G_3}^{Ake}(A) - Adv_{G_2}^{Ake}(A) \right| \leq q_{se}/2^d + q_{ro}/q^{nk}$.

G_4 : G_4 is the same as G_3 , except that in $H_l(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_4, k_2')$, the query executes a random response without checking the consistency of its output with the Execute query. That is, the Testexecksk (A, i, B, j, S) event does not occur in G_4 . \square

Lemma 4. For any polynomial adversary, the advantage of distinguishing G_4 and G_3 is $\left| Adv_{G_4}^{Ake}(A) - Adv_{G_3}^{Ake}(A) \right| \leq 2Adv_{R_q}^{MPWE}(t', q_{ro}) + 2Adv_{R_q}^{DMLWE}(t', q_{ro})$.

Proof. Clearly, if the Testexecksk (A, i, B, j, S) event does not occur, G_4 and G_3 are indistinct. If an adversary can cause Testexecksk (A, i, B, j, S) to occur with non-negligible probability, then a simulator can construct an algorithm D to solve the MPWE problem by running the adversary on G_3 . Given (A, X, Y, σ) , the algorithm simulates the game G_3 by changing it as follows.

1. When adversary makes the Execute $(\Pi_A^i, \Pi_B^i, \Pi_S^i)$ query, the algorithm D sets $b_{A0} = X + As_{A0} + e_{A0}$, $b_{B0} = Y + A^T s_{B0} + e_{B0}$, where $s_{A0}, e_{A0}, s_{B0}, e_{B0}$ are taken from the distribution β_{η}^k . At the same time, it is assumed that the adversary knows what is selected randomly and uniformly. This assumption will only increase the advantage of the adversary's successful attack.
2. When the adversary finishes, for each $H_l(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_4, k_2')$ query, where (b_{A0}, b_{B0}, n_1) is obtained in the Execute query, $k_2' = Rec(2v_2', \sigma_2) = \lfloor \bar{v}_2 \rfloor_{2q, 2}$, $\bar{v}_2 \leftarrow dbl(v_2)$, $v_2 = b_{A0}^T s_{B0} + e_{B2}$. Then, the algorithm D can compute:

$$\begin{aligned} v_2 &= b_{A0}^T s_{B0} + e_{B0} \\ &= (X + As_{A0} + e_{A0})^T (s_y + s_{B0}) + e_{B0} \\ &= X^T s_y + X^T s_{B0} + s_{A0}^T A^T s_y + s_{A0}^T A^T s_{B0} + e_{A0}^T s_y + e_{A0}^T s_{B0} + e_{B0} \\ &\approx X^T s_y + s_{A0}^T Y + (X + As_{A0} + e_{A0})^T s_{B0} \end{aligned} \quad (5)$$

Calculate $K = Rec(2X^T s_y, \sigma) = Rec\left(2\left(v_2 - s_{A0}^T Y - (X + As_{A0} + e_{A0})^T s_{B0}\right), \sigma\right)$ and add K to the list of possible values of the MPWE problem, at which point the MPWE problem is solved.

When the algorithm D simulates the game G_3 , algorithm D sets $b_{A0} = X + As_{A0} + e_{A0}$, $b_{B0} = Y + A^T s_{B0} + e_{B0}$ to replace the actual $b_{A0} = As_{A0} + e_{A0}$, $b_{B0} = A^T s_{B0} + e_{B0}$, respectively. Because X is randomly uniformly drawn from the distribution R_q^k , b_{A0} set by the algorithm and the actual b_{A0} are indistinguishable. Because $Y = A^T s_y + e_y$, unless the adversary can solve the DMLWE problem with a non-negligible advantage, the algorithm sets b_{B0} and the actual b_{B0} is indistinguishable. Considering the difficulty of the MPWE problem, assuming t' is the algorithm D 's running time, D creates a list of size q_{ro} with the advantage of ϵ , and $t' = O(t + (q_{se} + q_{ex} + q_{re} + q_{co})t_{exp})$.

For probabilistic polynomial adversaries A , the advantage of distinguishing between games G_4 and G_3 games is $\left| Adv_{G_4}^{Ake}(A) - Adv_{G_3}^{Ake}(A) \right| \leq 2Adv_{R_q}^{MPWE}(t', q_{ro}) + 2Adv_{R_q}^{DMLWE}(t', q_{ro})$.

G_5 : G_5 is the same as G_4 , unless the adversary is able to execute the Correctsk event before the Corrupt query. When the adversary executes the Correctsk event, the protocol terminates and the adversary succeeds. Compared to G_4 , G_5 makes following changes:

Before the Corrupt query, in $A1$ query to the user instance Π_A^i , if the Testsk! (A, i, B, S, l) event occurs, the protocol terminates and the adversary succeeds.

Before the Corrupt query, for the $H_l(\cdot)$ query, if the Testsk* (B, j, A, S) event occurs, the protocol terminates and the adversary succeeds. \square

Lemma 5. For any polynomial adversary, $Adv_{G_4}^{Ake}(A) \leq Adv_{G_5}^{Ake}(A)$.

Proof. Clearly, the definition only increases the advantage of the adversary; then, for any polynomial, the adversary $Adv_{G_4}^{Ake}(A) \leq Adv_{G_5}^{Ake}(A)$.

G_6 : G_6 is the same as G_5 unless the adversary guesses the password of the paired two user instances. At this point, the protocol is terminated and the adversary fails. If the Pairedskguess event occurs, the protocol terminates and the adversary fails. This section assumes that the test for Correctsk occurs after the test for Pairedskguess when the query is made.

This will make the following changes to G_5 : If a Testsk (A, i, B, S, l) event occurs for $l \in \{1, 2\}$ (this event should be checked in the $A1$ query or $H_l(\cdot)$ query), check whether the Testsk (A, i, B, j, S) event also occurs. \square

Lemma 6. For any polynomial adversary, the advantage of distinguishing G_6 and G_5 is $\left| Adv_{G_6}^{Ake}(A) - Adv_{G_5}^{Ake}(A) \right| \leq 2Adv_{R_q}^{MPWE}(t', q_{ro}) + 2Adv_{R_q}^{DMLWE}(t', q_{ro})$.

Proof. Clearly, G_6 and G_5 are indistinguishable if the Pairedskguess event does not occur. If the adversary can make the Pairedskguess occur with non-negligible probability in G_5 , then the algorithm D can be constructed to solve the MPWE problem by running the adversary in G_5 . Given (A, X, Y, σ) , the algorithm D simulates the game G_5 by changing the following.

In $A0$ queries to user instances Π_A^i , the algorithm D sets $b_{A0} = X$.

In the $B1$ query with input as $(TID_A, TID_B, n_1, n_2, b_{A0}, b_{S1})$ to the user instance Π_B^j , there is $A0$ query for the user instance Π_B^j with output $(TID_A, TID_B, n_1, b_{A0})$, and $S1$ query with input is $(TID_A, TID_B, n_1, b_{A0})$, and the output is $(TID_A, TID_B, n_1, n_2, b_{A0}, b_{S1})$, and set $b_{B0} = Y + A^T s_{B0} + e_{B0}$, where s_{B0}, e_{B0} are all taken from β_{η}^k .

In the $A1$ query to user instance Π_A^i , if Π_A^i has not been paired, the algorithm outputs 0 and aborts.

After the adversary attack is finished, for each $H_l(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_4, k_2')$ query, when b_{A0} and b_{B0} are in a related Π_A^i query, the algorithm D can be calculated as follows:

$$\begin{aligned} v_2 &= b_{A0}^T s_{B1} + e_{B1} \\ &= X^T (s_y + s_{B0}) + e_{B1} \\ &= X^T s_y + X^T s_{B0} + e_{B1} \\ &\approx X^T s_y + X^T s_{B0} \end{aligned} \quad (6)$$

Calculate $K = Rec(2X^T s_y, \sigma) = Rec(2(v_2 - X^T s_{B0}), \sigma)$, and add K to the list of possible values of the MPWE problem; then, the MPWE problem is solved.

When the algorithm D simulates the game G_5 , algorithm D sets $b_{B0} = Y + A^T s_{B0} + e_{B0}$ to replace actual $b_{B0} = A^T s_{B0} + e_{B0}$. Because $Y = A^T s_y + e_y$, unless the adversary can solve the DMLWE problem with non-negligible advantage, the algorithm D sets b_{B0} and the actual b_{B0} is indistinguishable. Considering the difficulty of the MPWE problem, assuming t' is the algorithm D 's running time, D creates a list of size q_{ro} with the advantage of ϵ , and $t' = O(t + (q_{se} + q_{ex} + q_{re} + q_{co})t_{exp})$.

For probabilistic polynomial adversaries A , the advantage of distinguishing between games G_6 and G_5 games is $\left| Adv_{G_6}^{Ake}(A) - Adv_{G_5}^{Ake}(A) \right| \leq 2Adv_{R_q}^{MPWE}(t', q_{ro}) + 2Adv_{R_q}^{DMLWE}(t', q_{ro})$.

G_7 : G_7 is the same as G_6 , except that there is an internal password oracle that keeps all passwords and is used to check the correctness of a given password in G_7 . This oracle is password-safe. The password oracle initializes all passwords and is unavailable to arbitrary polynomial adversaries.

The oracle accepts queries of the form $testpw(U, PW)$ and returns TRUE if $PW = PW_U$, FALSE otherwise. It also accepts a $Corrupt(U)$ query and returns $PW_1 = H_0(ID_i || PW_0)$ if U is a server, else returns PW_U . When a protocol receives a $Corrupt(U)$ query, it answers with a $Corrupt(U)$ query to the password oracle. \square

Lemma 7. For any polynomial adversary, G_7 and G_6 are indistinguishable, $Adv_{G_7}^{Ake}(A) = Adv_{G_6}^{Ake}(A)$.

Proof. Clearly, G_7 and G_6 are completely indistinguishable.

Now this section analyzes the advantages of the adversarial attack game G_7 . According to the definition of the game G_7 , this section can easily bound the probability of the adversary's success in the game as follows.

$$\Pr[Succ_7(A)] \leq \frac{\Pr[Correctpw] + (\Pr[Correctsk \neg Correctpw] + \Pr[Succ_7(A) \neg Correctsk \cap \neg Correctpw]) \cdot \Pr[\neg Correctpw]}{\Pr[\neg Correctsk \cap \neg Correctpw]} \cdot \Pr[\neg Correctpw]}{\Pr[\neg Correctsk \cap \neg Correctpw]} \quad (7)$$

For $\Pr[Correctpw]$, since passwords are randomly selected from a dictionary of size $|D|$ and will occur at most q_{se} queries to the password oracle, then $\Pr[Correctpw] \leq q_{se}/|D|$.

For $\Pr[Correctsk \neg Correctpw]$, since the adversary cannot decrypt n_4 without correctly guessing the password, the probability of the event $Correctsk$ is negligible; that is, the probability $\Pr[Correctsk \neg Correctpw]$ is negligible.

For $\Pr[Succ_7(A) \neg Correctsk \cap \neg Correctpw]$, the $Correctsk$ event and $Correctpw$ event have not occurred; then, if and only if the adversary successfully guesses the password used in the Test query on a fresh instance, the adversary succeeds in the attack. Since $\Pr[Correctsk | \neg Correctpw]$ is negligible, the $\Pr[\neg Correctsk \cap \neg Correctpw]$ probability is close to 1, so $\Pr[Succ_7(A) \neg Correctsk \cap \neg Correctpw] \cdot \Pr[\neg Correctsk \cap \neg Correctpw] \leq 1/2$.

In summary, $\Pr[Succ_7(A)] \leq q_{se}/|D| + 1/2(1 - q_{se}/|D|) \leq 1/2 + q_{se}/2|D|$, so for any polynomial adversary, the advantage of its attack game G_7 is $Adv_{G_7}^{Ake}(A) \leq q_{se}/|D|$.

Integrating G_0 to G_7 , the advantage of the adversary's successful attack is $Adv_{\Pi}^{Ake}(A) \leq (q_{ro} + q_{se} + q_{ex})^2/q^{nk} + q_{se}/2^d + q_{ro}/q^{nk} + 2Adv_{R_q}^{MPWE}(t', q_{ro}) + 2Adv_{R_q}^{DMLWE}(t', q_{ro}) + q_{se}/|D|$ and its value is negligible. The attacker bases the query on the random oracle, and the advantage of a successful attack is almost zero. The PAKE protocol scheme in this paper is provably secure based on the random oracle model, and the security of the protocol can ultimately be attributed to the difficulty of the MLWE problem on the lattice. \square

5.2. Security Properties

This section will mainly analyze how the three-party PAKE protocol of this paper satisfies the proposed security requirements.

1. Mutual authentication between three parties

In the proposed protocol, there is implicit authentication with user Π_A^i and explicit authentication with user Π_B^i for server Π_S^i , explicit authentication with user Π_B^i and server Π_S^i for user Π_A^i , and implicit authentication with user Π_A^i and server Π_S^i for Π_B^i . After the user Π_A^i obtains the shared key with users Π_B^i , the user Π_B^i can explicitly authenticate user Π_A^i 's identity by sending the authentication information or encrypting the message with

the shared key. Therefore, the proposed protocol realizes mutual authentication among the three parties, and only the legitimate party with a legitimate password can authenticate.

2. Known key security

The session keys are independent of each other. Even if the adversary obtains a specific session key, it cannot obtain other session keys through this session key. The final session key in this paper is $sk_A = H_2(TID_A, TID_B, b_{A0}, b_{B0}, n_1, n_4, k_2')$, which is constructed by the identity of the polynomial matrix regenerated each time b_{A0}, b_{B0} , random numbers and the secret information k_2' are exchanged and calculated by two users. Therefore, each session key is independent of each other, and the collision probability is minimal. Therefore, the three-party PAKE protocol proposed in this paper has known-key security.

3. Forward security

The forward security of PAKE means that even if the adversary obtains one or more shared passwords, the attacker cannot obtain the previously established session key. That is, session keys and passwords are independent of each other. Since the three-party PAKE protocol proposed in this paper needs the random number n_1, n_4 generated in this session, the calculated k_2' every time the session key is generated, and the session key is independent of the user password. Therefore, the protocol in this paper can provide forward security.

4. Resist three types of dictionary attacks

The dictionary attacks on the PAKE protocol can be divided into three categories: offline dictionary attacks, testable online dictionary attacks, and untestable online dictionary attacks.

For the offline dictionary attack, the adversary intercepts the information through the open channel and can carry out the dictionary attack $Auth_0, Auth_2, c_0$ and c_1 . If the adversary conducts dictionary attack on $Auth_0$ or c_0 , the adversary needs to overcome the MPWE problem with inputs b_{B0} and b_{S1} to calculate k_1 . If the adversary conducts a dictionary attack on $Auth_2$ or c_1 , it needs to overcome the MPWE problem with inputs b_{A0} and b_{S0} to calculate k_0 . According to the difficulty of the MPWE problem, the protocol in this paper can resist offline dictionary attacks.

For the measurable online dictionary attack, the adversary pretends to be the user Π_B^i and the server Π_S^i to launch an online dictionary attack on the user Π_A^i . Since the user Π_A^i will verify whether the server Π_S^i holds the password and whether the user Π_B^i is legal during the stage A1, once the verification fails many times, the user Π_A^i will realize that Π_A^i is the target of online dictionary attacks. Suppose the adversary pretends to be the user Π_A^i and the server Π_S^i to launch an online dictionary attack on the user Π_B^i . Then, user Π_B^i will authenticate user Π_A^i through the verification information or the message encrypted with the shared key after the key exchange is completed sent by the user Π_A^i . In that case, once the verification fails many times, user Π_B^i will realize that Π_B^i is the target of online dictionary attacks. Suppose the adversary pretends as user Π_A^i and user Π_B^i to launch an online dictionary attack on the server. Since the server Π_S^i can verify whether the user Π_B^i is legal in the stage S1, the verification fails many times. In that case, the server Π_S^i realizes it has become the target of the online dictionary attack. Once a protocol participant realizes that it has become the target of an online dictionary attack, it can immediately notify the other party that shares the password to update the password. Therefore, the protocol in this paper can resist measurable online dictionary attacks.

For untestable online dictionary attacks, since all online dictionary attacks in this protocol are detectable, the adversary cannot conduct untestable online dictionary attacks on the protocol participants.

6. Performance Analysis

In this section, the parameter settings of the proposed tripartite PAKE scheme and the comparison tripartite PAKE scheme are given, and the performance analysis is carried out from three aspects: computational efficiency, communication efficiency, and security. Considering that the computational complexity of symmetric encryption, hash function, and Peikert error reconciliation operation are much smaller than that of polynomial multi-

plication operation and public and private key generation operation, when comparing the computational efficiency, this section mainly considers polynomial multiplication operation and public and private key generation operation with high computational cost, and ignores the rest of the operations.

6.1. Parameter Selection

This section assumes that for all PAKE protocols, the size of the username is 64 bytes, the output length of the hash function is 256 bits, and the size of the random number is taken to be 256 bits. The parameters of the scheme implemented based on the RLWE hard problem are selected, and the parameters of the scheme based on the MLWE hard problem are set, as where the dimension of the polynomial in the polynomial ring, and the parameter of the central binomial distribution is the modulus $(n, \eta, q) = (768, 2, 3329)$ $(n, k, \eta, q) = (256, 3, 2, 3329)$ $nR_qk\eta q$ LWE_ESTIMATOR [33] for the LWE hard problem and the scheme based on the MLWE hard problem, respectively. The results show that both parameter configurations can achieve 222 bits of post-quantum security.

6.2. Computational Efficiency

The software implementation of the protocol presented in this paper was executed on a 64-bit system computer with 2.30 GHz 11th Gen Intel(R) Core(TM) i7-11800H and 16 GB RAM on Windows 11 Version 22H2. In order to improve the computational efficiency, the NTT algorithm was introduced $O(n^2)$ to reduce the $O(n \log n)$ computational complexity of polynomial multiplication and public and private key generation operations to a minimum. The following Table 1 provides the average operation time costs of polynomial multiplication and public and private key generation operations obtained from 1,000,000 measurements under different parameter configurations. It can be seen from the table that under the same post-quantum security bits, the average operation time of polynomial multiplication with MLWE parameters and RLWE parameters is almost the same, and the public and private key generation with MLWE parameters is slightly more expensive than that with RLWE parameters.

Table 1. Operation time of polynomial multiplication operation and public and private key generation operations.

Operation	Average Operation Time (ms) with MLWE Parameters	Average Operation Time (ms) with RLWE Parameters
Polynomial multiplication operation	0.05337	0.05732
Public and private key generation operations	0.11688	0.07364

6.3. Performance Comparison

Table 2 shows the comparison of the proposed tripartite PAKE scheme [21,34,35]. In terms of security, all the schemes can resist the offline dictionary attack. Compared with Choi's scheme [34], in terms of communication overhead, the amount of communication data of the proposed scheme is basically the same as that of other schemes [34,35]. In terms of computational overhead, the proposed scheme is almost the same as Liu's scheme. In addition, the tripartite PAKE protocol implemented based on the RLWE hard problem introduces a structured lattice which brings new security threats, while the proposed scheme implemented on the MLWE hard problem has higher potential security than the tripartite PAKE protocol implemented to solve the RLWE hard problem.

Table 2. Comparison of cost and security features.

Protocol	Choi Scheme [34]	Wang Scheme [21]		Liu's Scheme [35]	Shu Scheme [25]	Scheme of This Paper
		Implicit Authentication	Explicit Authentication			
Three-party mutual authentication	no	yes	yes	yes	yes	yes
Offline dictionary attack	yes	yes	yes	yes	yes	yes
Undetectable online dictionary attack	no	no	no	yes	yes	yes
Difficult hypothesis	RLWE		RLWE	RLWE	RLWE	MLWE
Public key length/byte	1536		1536	1536	1536	1536
Communication overhead/byte	9504	14,144	11,104	15,870	9696	10,080
Compute overhead/ms	0.63848	0.39228	0.39228	0.58116	0.75312	0.78774
Correspondence rounds	6	4	5	7	6	4
Error reconciliation mechanism	DING type	DING Pose	DING Pose	Peikert style	Peikert style	Peikert style
Security model	BPR model	BPR model	BPR model	BPR model	UC model	BPR model

7. Conclusions

The two-party PAKE protocol will bring huge overhead and management problems when applied to a large number of user communication scenarios. Therefore, researchers have proposed the three-party PAKE protocol. However, the current three-party PAKE protocols are often designed based on traditional difficult mathematical problems, which are vulnerable to quantum computing attacks. Therefore, it is urgent to study the three-party PAKE protocols that can resist quantum computing attacks. Based on the MLWE problem, this paper proposes a provably secure tripartite PAKE protocol under the BPR model for the first time. The security analysis shows that the proposed tripartite PAKE protocol realizes mutual authentication between three parties and can resist three types of dictionary attacks, and has higher or equivalent security than the existing schemes. Compared with the existing schemes, the proposed tripartite PAKE protocol has almost the same amount of communication data, but has the lowest communication rounds. Computational cost analysis shows that the computational cost of the proposed tripartite PAKE protocol is almost the same as that of the existing tripartite PAKE protocol. Considering the potential security problems of the tripartite PAKE protocol based on e RLWE hard problem, the tripartite PAKE protocol based on the MLWE hard problem proposed in this paper has higher practical value.

Author Contributions: Conceptualization, S.G. (Songhui Guo) and Y.S.; methodology, S.G. (Songhui Guo) and Y.S.; software, S.G. (Song Guo); validation, Y.Y. and S.S.; formal analysis, Y.S.; writing—original draft preparation, S.G. (Songhui Guo) and Y.S.; writing—review and editing, Y.Y. and S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are available on request to the authors.

Acknowledgments: The authors would like to thank anonymous reviewers for their valuable comments, which helped improve the content, organization, and quality of this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134.
2. Law, L.; Menezes, A.; Qu, M.; Solinas, J.; Vanstone, S.; Vanstone, S. An Efficient Protocol for Authenticated Key Agreement. *Des. Codes Cryptogr.* **2003**, *28*, 119–134. [\[CrossRef\]](#)
3. Abdalla, M.; Fouque, P.A.; Pointcheval, D. Password-Based Authenticated Key Exchange in the Three-Party Setting. In Proceedings of the International Conference on Theory & Practice in Public Key Cryptography, Les Diablerets, Switzerland, 23–26 January 2005.
4. Dongna, E.; Cheng, Q.; Ma, C. Password authenticated key exchange based on RSA in the three-party settings. In Proceedings of the Provable Security: Third International Conference, ProvSec 2009, Guangzhou, China, 11–13 November 2009; pp. 168–182, Proceedings 3.
5. Lin, C.; Sun, H.; Hwang, T. Three-party encrypted key exchange: Attacks and a solution. *ACM SIGOPS Oper. Syst. Rev.* **2000**, *34*, 12–20. [\[CrossRef\]](#)
6. Chang, T.; Hwang, M.; Yang, W. A communication-efficient three-party password authenticated key exchange protocol. *Inf. Sci.* **2011**, *181*, 217–226. [\[CrossRef\]](#)
7. Steiner, M.; Tsudik, G.; Waidner, M. Refinement and extension of encrypted key exchange. *ACM SIGOPS Oper. Syst. Rev.* **1995**, *29*, 22–30. [\[CrossRef\]](#)
8. Ding, Y.; Horster, P. Undetectable on-line password guessing attacks. *ACM SIGOPS Oper. Syst. Rev.* **1995**, *29*, 77–86. [\[CrossRef\]](#)
9. Lin, C.; Sun, H.; Steiner, M.; Hwang, T. Three-party encrypted key exchange without server public-keys. *IEEE Commun. Lett.* **2001**, *5*, 497–499. [\[CrossRef\]](#)
10. Lee, T.; Hwang, T.; Lin, C. Enhanced three-party encrypted key exchange without server public keys. *Comput. Secur.* **2004**, *23*, 571–577. [\[CrossRef\]](#)
11. Lu, R.; Cao, Z. Simple three-party key exchange protocol. *Comput. Secur.* **2007**, *26*, 94–97. [\[CrossRef\]](#)
12. Huang, H.F. A simple three-party password-based key exchange protocol. *Int. J. Commun. Syst.* **2009**, *22*, 857–862. [\[CrossRef\]](#)
13. Lee, C.; Li, C.; Hsu, C. A three-party password-based authenticated key exchange protocol with user anonymity using extended chaotic maps. *Nonlinear Dyn.* **2013**, *73*, 125–132. [\[CrossRef\]](#)
14. Zhao, J.; Gu, D. Provably secure three-party password-based authenticated key exchange protocol. *Inf. Sci.* **2012**, *184*, 310–323. [\[CrossRef\]](#)
15. Lou, D.C.; Huang, H.F. Efficient three-party password-based key exchange scheme. *Int. J. Commun. Syst.* **2011**, *24*, 504–512. [\[CrossRef\]](#)
16. Wu, S.; Chen, K.; Zhu, Y. Enhancements of a three-party password-based authenticated key exchange protocol. *Int. Arab. J. Inf. Technol.* **2013**, *10*, 215–221.
17. Mao, Y. Password Authenticated Key Exchange Protocol in the Three Party Setting Based on Lattices. *J. Electron. Inf. Technol.* **2014**, *35*, 1376–1381.
18. Katz, J.; Vaikuntanathan, V. *Smooth Projective Hashing and Password-Based Authenticated Key Exchange from Lattices*; Springer: Berlin/Heidelberg, Germany, 2009.
19. Xu, D.; He, D.; Choo, K.R.; Chen, J. Provably secure three-party password authenticated key exchange protocol based on ring learning with error. *Cryptol. ePrint Arch.* **2017**.
20. Ding, J.; Alsayigh, S.; Lancrenon, J.; Saraswathy, R.V.; Snook, M. Provably Secure Password Authenticated Key Exchange Based on RLWE for the Post-Quantum World. In Proceedings of the Cryptographers Track at the RSA Conference, San Francisco, CA, USA, 14–17 February 2017.
21. Wang, C.; Chen, L. Three-party password authenticated key agreement protocol with user anonymity based on lattice. *J. Commun.* **2018**, *39*, 21–30.
22. Yu, J.; Lian, H.; Tang, Y.; Shi, M.; Zhao, Z. Password-based three-party authenticated key exchange protocol from lattices. *J. Commun.* **2018**, *39*, 87–97.
23. Zhang, J.; Yu, Y. Two-round PAKE from approximate SPH and instantiations from lattices. In Proceedings of the Advances in Cryptology—ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; pp. 37–67, Proceedings, Part III 23.
24. Gao, X.; Ding, J.; Liu, J.; Li, L. Post-quantum secure remote password protocol from RLWE problem. In Proceedings of the Information Security and Cryptology: 13th International Conference, Inscrypt 2017, Xi’an, China, 3–5 November 2017; pp. 99–116, Revised Selected Papers 13.
25. Shu, Q.; Wang, S.; Hu, B.; Han, L. Verifier-Based Three-Party Password-Authenticated Key Exchange Protocol from Ideal Lattices. *J. Cryptol. Res.* **2021**, *8*, 294–306. [\[CrossRef\]](#)
26. Bai, S.; Lepoint, T.; Roux-Langlois, A.; Sakzad, A.; Stehlé, D.; Steinfeld, R. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. *J. Cryptol.* **2015**, *31*, 610–640. [\[CrossRef\]](#)
27. Langlois, A.; Stehlé, D. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.* **2015**, *75*, 565–599. [\[CrossRef\]](#)

28. Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018; pp. 353–367.
29. Ding, J.; Lin, X. A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. *Iacr Cryptol. Eprint Arch.* **2013**.
30. Peikert, C. Lattice cryptography for the internet. In Proceedings of the Post-Quantum Cryptography: 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, 1–3 October 2014; pp. 197–219, Proceedings 6.
31. Bellare, M.; Pointcheval, D.; Rogaway, P. Authenticated Key Exchange Secure Against Dictionary Attacks. In Proceedings of the International Conference on the Theory & Applications of Cryptographic Techniques, Bruges, Belgium, 14–18 May 2000.
32. Bellare, M.; Rogaway, P. Entity authentication and key distribution. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 22–26 August 1993; pp. 232–249.
33. Albrecht, M.R.; Player, R.; Scott, S. On the concrete hardness of Learning with Errors. *J. Math. Cryptol.* **2015**, *9*, 169–203. [[CrossRef](#)]
34. Choi, R.; An, H.; Kim, K. AtLast: Another three-party lattice-based PAKE scheme. In Proceedings of the 2018 Symposium on Cryptography and Information Security (SCIS 2018), Niigata, Japan, 23–26 January 2018.
35. Liu, C.; Zheng, Z.; Jia, K.; You, Q. Provably secure three-party password-based authenticated key exchange from RLWE. In Proceedings of the Information Security Practice and Experience: 15th International Conference, ISPEC 2019, Kuala Lumpur, Malaysia, 26–28 November 2019; pp. 56–72; Proceedings 15.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.